

ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ

Токарева О.В.
АлтГПА, 2014

АЛГЕБРА ЛОГИКИ И ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

Логические парадоксы

- Парадокс Рассела (1901 год)

Пусть K — множество всех множеств, которые не содержат себя в качестве своего элемента. Содержит ли K само себя в качестве элемента? Если предположить, что содержит, то мы получаем противоречие с "Не содержат себя в качестве своего элемента". Если предположить, что K не содержит себя как элемент, то вновь возникает противоречие, ведь K — множество всех множеств, которые не содержат себя в качестве своего элемента, а значит должно содержать все возможные элементы, включая и себя.

Варианты формулировок

- Наиболее ранняя из формулировок, приписываемая софистам, называется **Парадокс лжеца**: «Критянин сказал, что все критяне лжецы. Сказал ли он правду?»
- **Парадокс брадобрея**: Единственному деревенскому брадобрею приказали: *«Брить всякого, кто сам не бреется, и не брить того, кто сам бреется»*. Кто побреет брадобрея?

- В одной стране вышел указ: «*Мэры всех городов должны жить не в своем городе, а в специальном Городе мэров*». Где должен жить мэр Города мэров?
- Некая библиотека решила составить библиографический каталог, в который входили бы все те и только те библиографические каталоги, которые не содержат ссылок на самих себя. Должен ли такой каталог включать ссылку на себя?



Алгебра логики (булева алгебра) – это раздел математики, возникший в XIX в. благодаря усилиям английского математика [Дж. Буля](#).

Поначалу булева алгебра не имела никакого практического значения.

Однако уже в XX в. её положения нашли применение в описании функционирования и разработке различных электронных схем.

Законы алгебры логики стал использоваться при проектировании различных частей компьютеров (память, процессор).

Высказывание –

*это повествовательное предложение, о котором можно сказать **правда** или **ложь**.*

«Два больше одного» – истина

«5.8 является целым числом» – ложь

Алгебра логики не касается смысла высказываний



Как связываются между собой простые логические высказывания, образуя сложные? В естественном языке мы используем различные союзы и другие части речи. Например, «и», «или», «либо», «не», «если», «то», «тогда».

Примеры: «У него есть знания, **и** он умеет их применять», «Она приедет во вторник, **либо** в среду», «Я пойду в кино **тогда**, когда сделаю задание», «5 **не** равно 6», «**Если** мне позвонит друг, **то** я буду очень рад».

Логически, а иногда неосознанно, исходя из предыдущего жизненного опыта, мы понимаем, что правда при союзе «и» будет только в случае правдивости обоих простых высказываний. Стоит одному стать ложью, и всё сложное высказывание будет лживо.

А вот при связке «или» правдой может быть только одно простое высказывание, в этом случае все выражение станет истинным.

Булева алгебра переложила этот жизненный опыт на аппарат математики, формализовала его, ввела жесткие правила получения однозначного результата.

Алгебра логики предусматривает множество логических операций.

Три являются основными, т.к. с их помощью можно описать все остальные.

Это **конъюнкция (И)**, **дизъюнкция (ИЛИ)** и **инверсия (НЕ)**.

Таблицы истинности

Логические операции удобно описывать **таблицами истинности**, в которых отражают результаты вычислений сложных высказываний при различных значениях исходных простых высказываний.

Простые высказывания обозначаются переменными (например, **X** и **Y**).

x	y	not x	x or y	x and y
true	true	false	true	true
true	false	false	true	false
false	true	true	true	false
false	false	true	false	false

x	y	not x	x or y	x and y
1	1	0	1	1
1	0	0	1	0
0	1	1	1	0
0	0	1	0	0

В ЭВМ используются различные устройства, работу которых прекрасно описывает алгебра логики: группы переключателей, триггеры, сумматоры.

Кроме того, связь между булевой алгеброй и компьютерами лежит и в используемой в ЭВМ двоичной системе счисления.

Поэтому в устройствах компьютера можно хранить и преобразовывать как числа, так и значения логических переменных.

Переключательные схемы

В ЭВМ применяются электрические схемы, состоящие из множества переключателей. Переключатель может находиться только в двух состояниях: замкнутом и разомкнутом. В первом случае ток проходит, во втором нет. Описывать работу таких схем очень удобно с помощью алгебры логики. В зависимости от положения переключателей можно получить или не получить сигналы на выходах.

Вентили, триггеры и сумматоры

Вентиль представляет собой логический элемент, который принимает одни двоичные значения и выдает другие в зависимости от своей реализации.

Так, например, есть вентили, реализующие логическое умножение (конъюнкцию), сложение (дизъюнкцию) и отрицание (инверсию).

Триггеры и сумматоры – это сложные устройства, состоящие из более простых элементов – вентиляей.

Триггер способен хранить один двоичный разряд, за счет того, что может находиться в двух устойчивых состояниях.

В основном триггеры используется в регистрах процессора.

Сумматоры широко используются в арифметико-логических устройствах (АЛУ) процессора и выполняют суммирование двоичных разрядов.

Логические элементы вентили

Вентиль - это устройство, которое выдает результат булевой операции от введенных в него данных (сигналов).

Простейший вентиль представляет собой транзисторный инвертор, который преобразует низкое напряжение в высокое или наоборот (высокое в низкое).

Это можно представить как преобразование логического нуля в логическую единицу или наоборот. Т.е. получаем вентиль **НЕ**.

Соединив пару транзисторов различным способом, получают вентили **ИЛИ-НЕ** и **И-НЕ**.

Эти вентили принимают уже не один, а два и более входных сигнала.

Выходной сигнал всегда один и зависит (выдает высокое или низкое напряжение) от входных сигналов.

В случае вентиля ИЛИ-НЕ получить высокое напряжение (логическую единицу) можно только при условии низкого напряжении на всех входах.

В случае вентиля И-НЕ все наоборот: логическая единица получается, если все входные сигналы будут нулевыми.

Это обратное логическим операциям И и ИЛИ.

Однако обычно используются вентили И-НЕ и ИЛИ-НЕ, т.к. их реализация проще: И-НЕ и ИЛИ-НЕ реализуются двумя транзисторами, тогда как логические И и ИЛИ тремя.

Выходной сигнал вентиля можно выразить как функцию от входных.

Транзистору требуется очень мало времени для переключения из одного состояния в другое (время переключения оценивается в наносекундах).

И в этом одно из существенных преимуществ схем, построенных на их основе.

Обозначения

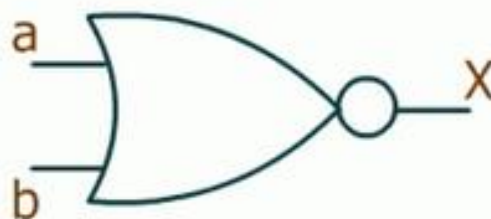
Таблицы истинности

НЕ



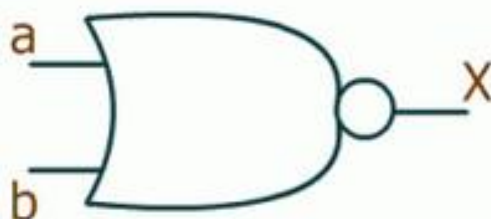
a	X
0	1
1	0

ИЛИ-НЕ



a	b	X
0	0	1
0	1	0
1	0	0
1	1	0

И-НЕ



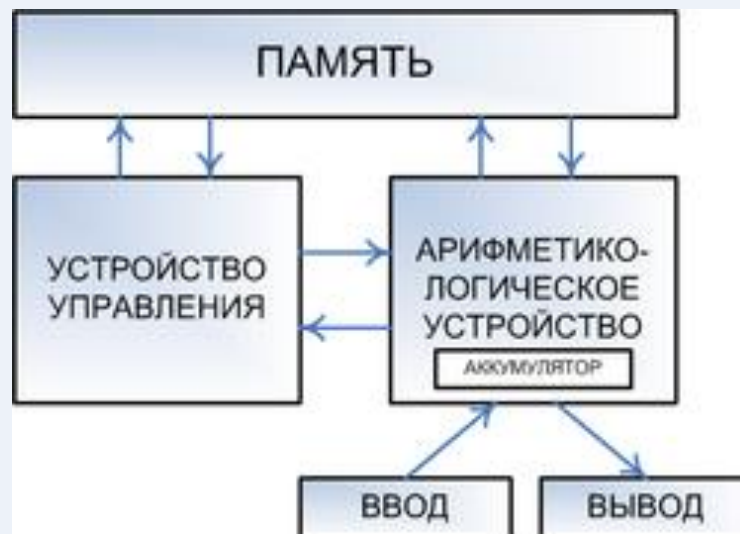
a	b	X
0	0	1
0	1	1
1	0	1
1	1	0

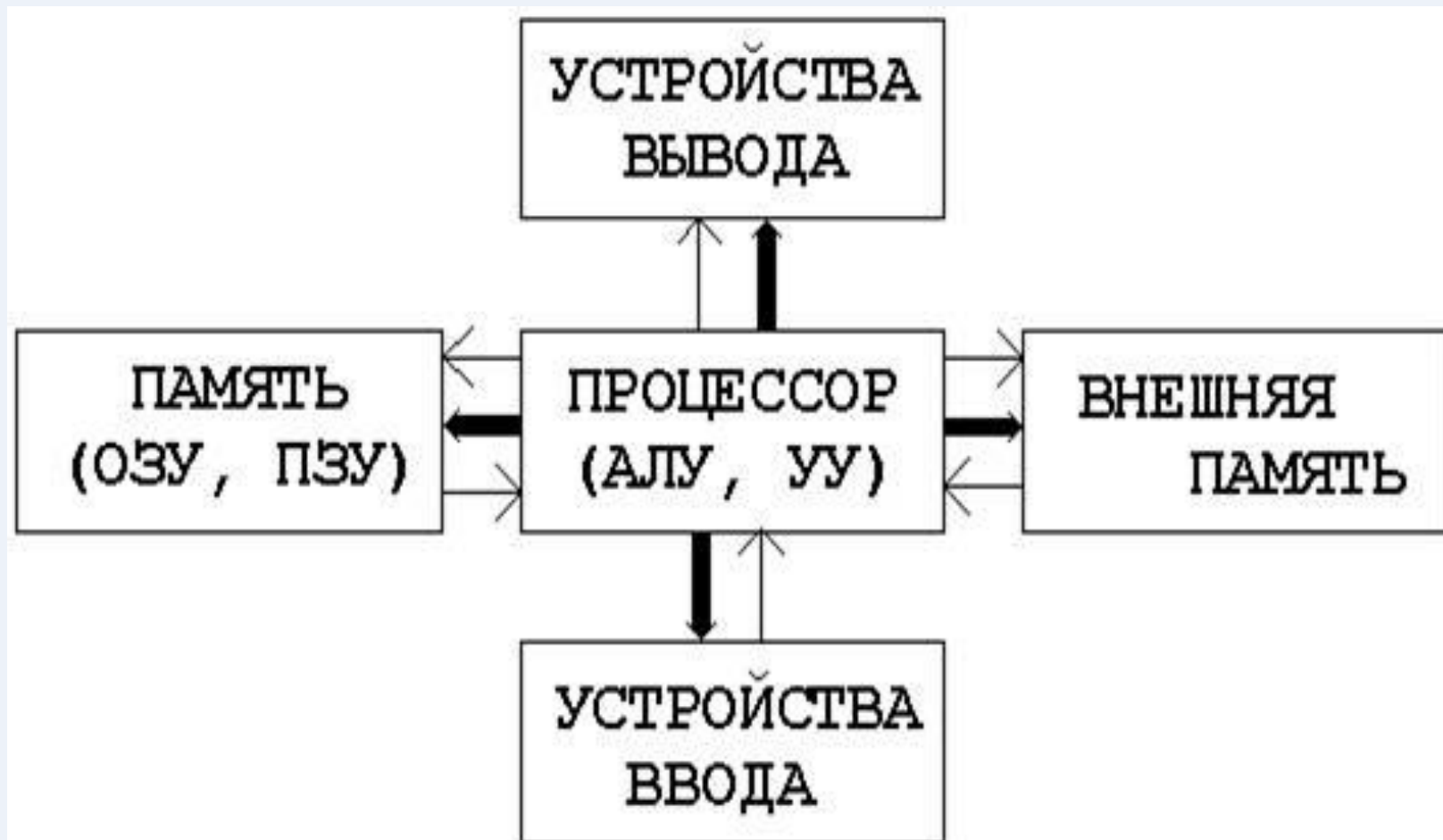
Схема фон-Неймана

В марте 1945 г. принципы логической архитектуры компьютера были оформлены в документе, который назывался «Первый проект отчёта о EDVAC» — отчет для Баллистической Лаборатории Армии США, на чьи деньги осуществлялась постройка ЭНИАКа и разработка EDVACa.

Архитектура фон Неймана — широко известный принцип совместного хранения команд и данных в памяти компьютера.

Вычислительные системы такого рода часто обозначают термином «машина фон Неймана», «схема фон Неймана».





Устройство
ввода

Устройство
вывода

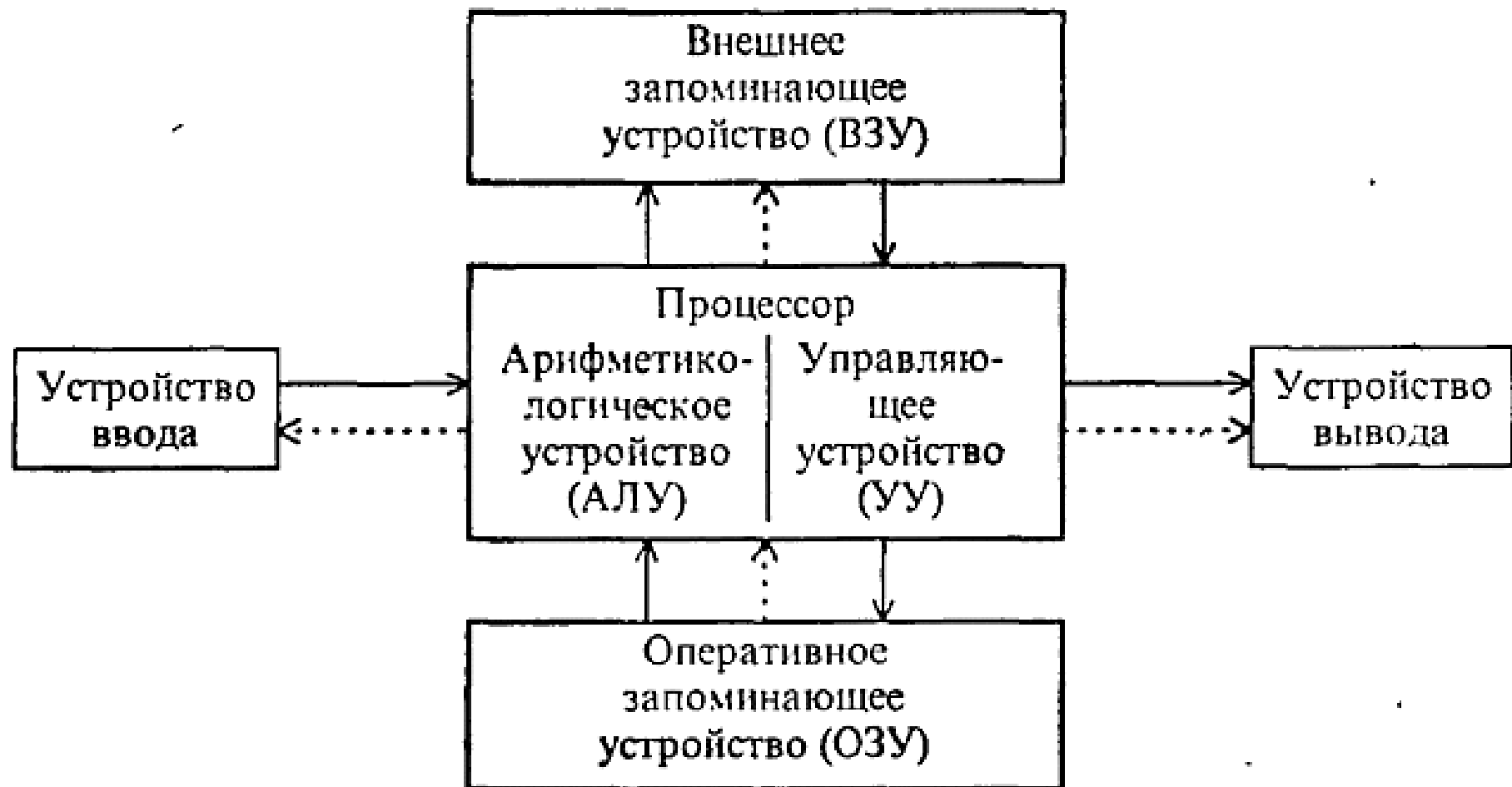
Устройство
ввода/вывода

Память

Арифметико-
логическое
устройство
ввода

Устройство
управления

Процессор



Принципы фон Неймана

- *Принцип однородности памяти*

Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы.

Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему.

Это позволяет производить над командами те же операции, что и над числами.

Например, циклически изменяя адресную часть команды, можно обеспечить обращение к последовательным элементам массива данных.

Такой прием носит название модификации команд и с позиций современного программирования не приветствуется.

Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы.

Эта возможность лежит в основе трансляции — перевода текста программы с языка высокого уровня на язык конкретной вычислительной машины.

- ***Принцип адресности***

Структурно основная память состоит из пронумерованных ячеек, причем процессору в произвольный момент доступна любая ячейка.

Двоичные коды команд и данных разделяются на единицы информации, называемые словами, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих ячеек — адреса.

- ***Принцип программного управления***

Все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд.

Каждая команда предписывает некоторую операцию из набора операций, реализуемых вычислительной машиной.

Команды хранятся в последовательных ячейках памяти вычислительной машины и выполняются в порядке их расположения в программе.

При необходимости, с помощью специальных команд, эта последовательность может быть изменена.

Решение об изменении порядка выполнения команд принимается либо на основании анализа результатов предшествующих вычислений, либо безусловно.

- ***Принцип двоичного кодирования***

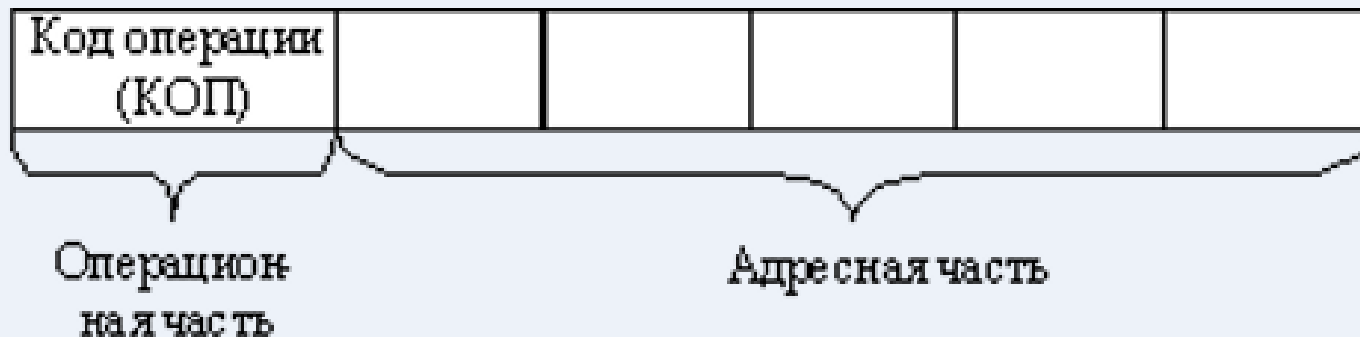
Согласно этому принципу, вся информация (как данные, так и команды) кодируется двоичными цифрами 0 и 1.

Каждый тип информации представляется двоичной последовательностью и имеет свой формат.

Последовательность битов в формате, имеющая определенный смысл, называется полем.

В числовой информации обычно выделяют поле знака и поле значащих разрядов.

В формате команды можно выделить два поля: поле кода операции и поле адресов.

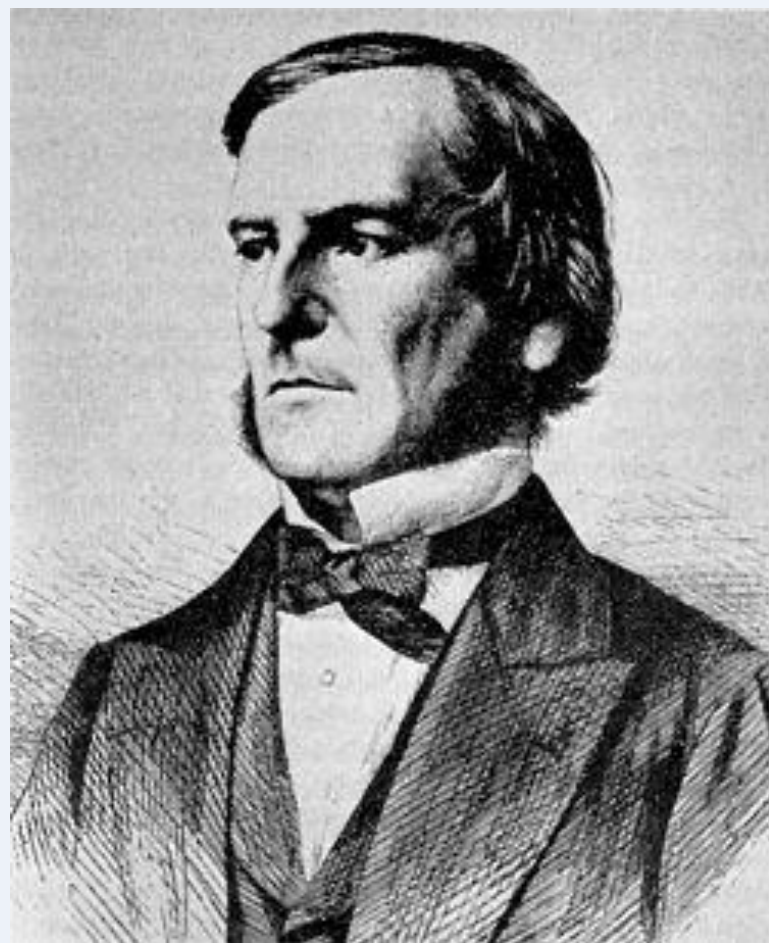


Джордж Буль (1815 – 1864)

George Boole – английский математик и логик.

В 12 лет знал латынь, позднее греческий, французский, немецкий и итальянский языки.

Зачитывался математическими журналами Механического института, интересовался работами математиков прошлого – Ньютона, Лапласа, Лагранжа, проблемами современной алгебры.



Начиная с 1839 г. Буль стал посылать свои работы в Кембриджский математический журнал.

После того как Буль убедился, что его алгебра вполне применима к логике, в 1847 г. он опубликовал памфлет «Математический анализ логики», в котором высказал идею, что логика более близка к математике, чем к философии.

В 1854 г. опубликовал работу «Исследование законов мышления, базирующихся на математической логике и теории вероятностей».

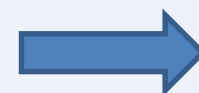
Работы 1847 и 1854 гг. положили начало алгебре логики, или булевой алгебре.

Буль первым показал, что существует аналогия между алгебраическими и логическими действиями, так как и те, и другие предполагают лишь два варианта ответов – истина или ложь (1 0).

Он придумал систему обозначений и правил, пользуясь которыми можно было закодировать любые высказывания, а затем манипулировать ими как обычными числами.

В своей работе «Законы мышления» (1854 г.) Буль окончательно сформулировал основы математической логики.

Также попытался сформулировать общий метод вероятностей, с помощью которого из заданной системы вероятных событий можно было бы определить вероятность последующего события, логически связанного с ними.

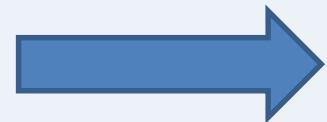


Джон фон Нейман (1903 – 1957)

Венгеро-американский математик, сделавший важный вклад в квантовую физику, квантовую логику, функциональный анализ, теорию множеств, информатику, экономику и др. науки.



Наиболее известен как человек, с именем которого связывают архитектуру большинства современных компьютеров (архитектура фон Неймана), применением теории операторов к квантовой механике (алгебра фон Неймана), а также как участник Манхэттенского проекта (программа США по разработке ядерного оружия) и как создатель теории игр и концепции клеточных автоматов.



<http://www.inf1.info/book/export/html/210>